i.MX8 SHE API Rev 0.1

Generated by Doxygen 1.8.17

1 SHE API

This document is a software referece description of the API provided by the i.MX8 SHE solutions.

2 Revision History

Revision	date	description
0.1	Jul 06 2023	first draft

3 General concepts related to the API

3.1 Session

The API must be initialized by a potential requestor by opening a session.
The session establishes a route (MU, DomainID...) between the requester and the SHE module, and grants the usage of a specified key store. When a session is opened, the SHE module returns a handle identifying the session to the requester.

4 Module Index

4.1 Modules

Here is a list of all modules:

5 Module Documentation

5.1 CMD_ENC_CBC / CMD_DEC_CBC and CMD_ENC_ECB / CMD_DEC_ECB

Macros

- #define **SHE_CIPHER_ONE_GO_ALGO_AES_ECB** ((she_op_cipher_one_go_algo_t)(0x00u))
- #define **SHE_CIPHER_ONE_GO_ALGO_AES_CBC** ((she_op_cipher_one_go_algo_t)(0x01u))
- #define SHE_CIPHER_ONE_GO_ALGO_AES_CCM ((she_op_cipher_one_go_algo_t)(0x04u))
- #define **SHE_CIPHER_ONE_GO_ALGO_SM4_ECB** ((she_op_cipher_one_go_algo_t)(0x10u))
- #define **SHE_CIPHER_ONE_GO_ALGO_SM4_CBC** ((she_op_cipher_one_go_algo_t)(0x11u))
- #define SHE_CIPHER_ONE_GO_FLAGS_DECRYPT ((she_op_cipher_one_go_flags_t)(0u << 0))
- #define SHE_CIPHER_ONE_GO_FLAGS_ENCRYPT ((she_op_cipher_one_go_flags_t)(1u << 0))

Typedefs

- typedef uint8_t she_op_cipher_one_go_algo_t
- typedef uint8_t she_op_cipher_one_go_flags_t

Functions

- she_err_t she_open_cipher_service (she_hdl_t session_hdl, open_svc_cipher_args_t ∗args)
- she_err_t she_close_cipher_service (she_hdl_t cipher_handle)
- she_err_t she_cipher_one_go (she_hdl_t cipher_handle, op_cipher_one_go_args_t ∗args)

5.1.1 Detailed Description

5.1.2 Macro Definition Documentation

5.1.2.1 SHE_CIPHER_ONE_GO_ALGO_AES_CCM `#define SHE_CIPHER_ONE_GO_ALGO_AES_CCM ((she_op_cipher_one_go_a`

Perform AES CCM with following constraints:

- AES CCM where: – Adata = 0, – Tlen = 16 bytes, – nonce size = 12 bytes

5.1.2.2 SHE_CIPHER_ONE_GO_FLAGS_DECRYPT `#define SHE_CIPHER_ONE_GO_FLAGS_DECRYPT ((she_op_cipher_one_go_`
`<< 0))`

Bit indicating the decrypt operation

5.1.2.3 SHE_CIPHER_ONE_GO_FLAGS_ENCRYPT `#define SHE_CIPHER_ONE_GO_FLAGS_ENCRYPT ((she_op_cipher_one_go_`
`<< 0))`

Bit indicating the encrypt operation

5.1.3 Typedef Documentation

5.1.3.1 she_op_cipher_one_go_algo_t `typedef uint8_t she_op_cipher_one_go_algo_t`

Bit field indicating the requested cipher operations

5.1.3.2 she_op_cipher_one_go_flags_t `typedef uint8_t she_op_cipher_one_go_flags_t`

Bit field indicating the requested encrypt/decrypt operations

5.1.4 Function Documentation

5.1.4.1 she_open_cipher_service()

```
she_err_t she_open_cipher_service (
        she_hdl_t session_hdl,
        open_svc_cipher_args_t ∗ args )
```

- Open a cipher service flow.
- User can call this function only after having opened a key-store service flow.
- User must open this service in order to perform cipher operation.

Parameters

session_hdl	handle identifying the SHE session.
args	pointer to the structure containing the function arguments.

Returns

error code.

5.1.4.2 she_close_cipher_service()

```
she_err_t she_close_cipher_service (
        she_hdl_t cipher_handle )
```

Terminate a previously opened cipher service flow

Parameters

cipher_handle	handle identifying the Cipher service.

Returns

error code.

5.1.4.3 she_cipher_one_go()

```
she_err_t she_cipher_one_go (
        she_hdl_t cipher_handle,
        op_cipher_one_go_args_t * args )
```

Perform ciphering operation i.e.

CBC encryption/decryption and ECB encryption/decryption of a given plaintext/ciphertext with the key identified by key_id.

User can call this function only after having opened a cipher service flow

Parameters

session_hdl	handle identifying the SHE session.
args	pointer to the structure containing the function arguments.

Returns

error code

5.2 CMD_EXPORT_RAM_KEY

Data Structures

- struct op_export_plain_key_args_t

Functions

- she_err_t she_export_plain_key (she_hdl_t utils_handle, op_export_plain_key_args_t ∗args)

5.2.1 Detailed Description

5.2.2 Data Structure Documentation

5.2.2.1 struct op_export_plain_key_args_t Structure describing the export RAM key operation arguments

Data Fields

uint8_t ∗	m1	< identifier of the key to be used for the operation pointer to the output address for M1 message
uint8_t	m1_size	size of M1 message - 128 bits
uint8_t ∗	m2	pointer to the output address for M2 message
uint8_t	m2_size	size of M2 message - 256 bits
uint8_t ∗	m3	pointer to the output address for M3 message
uint8_t	m3_size	size of M3 message - 128 bits
uint8_t ∗	m4	pointer to the output address for M4 message
uint8_t	m4_size	size of M4 message - 256 bits
uint8_t ∗	m5	pointer to the output address for M5 message
uint8_t	m5_size	size of M5 message - 128 bits

5.2.3 Function Documentation

5.2.3.1 she_export_plain_key()

```
she_err_t she_export_plain_key (
        she_hdl_t utils_handle,
        op_export_plain_key_args_t ∗ args )
```

exports the RAM_KEY into a format protected by SECRET_KEY.

Parameters

utils_handle	handle identifying the SHE utils service.
args	pointer to the structure containing the function arguments.

Returns

error code

5.3 FAST_MAC

Data Structures

- struct op_fast_seco_mac_t
- struct op_fast_v2x_mac_t

Macros

- #define SHE_FAST_MAC_FLAGS_GENERATION 0
- #define **SHE_FAST_MAC_FLAGS_VERIFICATION** 1
- #define **SHE_FAST_MAC_FLAGS_VERIF_BIT_LEN** 2

5.3.1 Detailed Description

5.3.2 Data Structure Documentation

5.3.2.1 struct op_fast_seco_mac_t Structure describing the fast mac generation operation arguments for S↩ ECO

Data Fields

uint16_t	key_id	identifier of the key to be used for the operation
uint16_t	data_length	length in bytes of the input message. The message is padded to be a multiple of 128 bits by SHE
uint16_t	data_offset	Offset of the Input data in the SECURE RAM.
uint8_t	mac_length	MAC length in bytes, only valid in case of MAC verification.
uint8_t	flags	flag to identify the operation(generate/verify)
uint32_t	verification_status	result of the MAC comparison

5.3.2.2 struct op_fast_v2x_mac_t Structure describing the fast mac generation operation arguments for V2X

Data Fields

uint16_t	key_id	identifier of the key to be used for the operation
uint16_t	data_length	length in bytes of the input message. The message is padded to be a multiple of 128 bits by SHE
uint16_t	rsrv	reserved
uint8_t	mac_length	MAC length expressed in bits, only valid in case of MAC verification. Accepted values are: Zero: the MAC length value used will be the nominal length (128bit). Greater or equal than the minimum value defined in the key store.
uint8_t	flags	flag to identify the operation(generate/verify)
uint32_t	m1	
uint32_t	m2	
uint32_t	m3	
uint32_t	m4	The message to use for MAC generation or verification.
uint32_t	verification_status	result of the MAC comparison

5.3.3 Macro Definition Documentation

5.3.3.1 SHE_FAST_MAC_FLAGS_GENERATION `#define SHE_FAST_MAC_FLAGS_GENERATION 0`

Macros to identify MAC operation type

5.4 CMD_GENERATE_MAC

Data Structures

- struct op_generate_mac_t
- struct op_get_id_args_t

Macros

- #define SHE_MAC_SIZE 16u

  size of the MAC generated is 128bits.
- #define SHE_CHALLENGE_SIZE 16u /∗ 128 bits ∗/

  size of the input challenge vector is 128 bits.
- #define SHE_ID_SIZE 15u /∗ 120 bits ∗/

  size of the Identity(ID) returned is 120 bits.
- #define SHE_MAC_SIZE 16u /∗ 128 bits ∗/

  size of the computed MAC is 128 bits.

Functions

- she_err_t she_generate_mac (she_hdl_t utils_handle, op_generate_mac_t ∗args)
- she_err_t she_get_id (she_hdl_t utils_handle, op_get_id_args_t ∗args)

5.4.1 Detailed Description

5.4.2 Data Structure Documentation

5.4.2.1 struct op_generate_mac_t Structure describing the fast mac generation operation arguments

Data Fields

uint16_t	key_ext	identifier of the key extension to be used for the operation
uint16_t	key_id	identifier of the key to be used for the operation
uint16_t	message_length	length in bytes of the input message. The message is padded to be a multiple of 128 bits by SHE
uint8_t ∗	message	pointer to the message to be processed
uint8_t ∗	mac	pointer to where the output MAC should be written (128bits should be allocated there)

5.4.2.2 struct op_get_id_args_t Structure describing the fast mac generation operation arguments for SECO

Data Fields

uint8_t	challenge[SHE_CHALLENGE_SIZE]	Challenge vector.
uint8_t	id[SHE_ID_SIZE]	identity (UID) returned by the command
uint8_t	sreg	status register returned by the command
uint8_t	mac[SHE_MAC_SIZE]	MAC returned by the command.

5.4.3 Function Documentation

5.4.3.1 she_generate_mac()

```
she_err_t she_generate_mac (
        she_hdl_t utils_handle,
        op_generate_mac_t * args )
```

Generates a MAC of a given message with the help of a key identified by key_id.

Parameters

utils_handle	handle identifying the utils service.
args	pointer to the structure containing the function arguments.

Returns

error code

5.4.3.2 she_get_id()

```
she_err_t she_get_id (
        she_hdl_t utils_handle,
        op_get_id_args_t * args )
```

This function returns the identity (UID) and the value of the status register protected by a MAC over a challenge and the data. User can call this function only after getting the utility service.

Parameters

utils_handle	handle identifying the utils service.
args	pointer to the structure containing the function arguments.

Returns

error code

5.5 CMD_VERIFY_MAC

Data Structures

- struct op_verify_mac_t

Macros

- #define **SHE_FAST_MAC_VERIFICATION_STATUS_OK** 0x5a3cc3a5
- #define **MAC_BYTES_LENGTH** 0
- #define **MAC_BITS_LENGTH** 1
- #define SHE_MAC_VERIFICATION_SUCCESS 0

  indication of mac verification success
- #define SHE_MAC_VERIFICATION_FAILED 1

  indication of mac verification failure

Functions

- she_err_t she_verify_mac (she_hdl_t utils_handle, op_verify_mac_t ∗args)

5.5.1 Detailed Description

5.5.2 Data Structure Documentation

5.5.2.1 struct op_verify_mac_t Structure describing the fast mac generation operation arguments

Data Fields

uint16_t	key_ext	identifier of the key extension to be used for the operation
uint16_t	key_id	identifier of the key to be used for the operation
uint16_t	message_length	length in bytes of the input message. The message is padded to be a multiple of 128 bits by SHE
uint8_t ∗	message	pointer to the message to be processed
uint8_t ∗	mac	pointer to the MAC to be compared
uint8_t	mac_length	number of MAC bytes to be compared with the expected value. It cannot be lower than 4 bytes.
uint32_t	verification_status	result of the MAC comparison
uint8_t	mac_length_encoding	

5.5.3 Function Documentation

5.5.3.1 she_verify_mac()

```
she_err_t she_verify_mac (
        she_hdl_t utils_handle,
        op_verify_mac_t ∗ args )
```

Verify the MAC of a given message with the help of a key identified by key_id.

Parameters

utils_handle	handle identifying the utils service.
args	pointer to the structure containing the function arguments.

Returns

error code

5.6 SHE get info

Functions

- she_err_t she_get_info (she_hdl_t session_hdl, op_get_info_args_t ∗args)

5.6.1 Detailed Description

5.6.2 Function Documentation

5.6.2.1 she_get_info()

```
she_err_t she_get_info (
        she_hdl_t session_hdl,
        op_get_info_args_t * args )
```

Perform device attestation operation Get miscellaneous information. This function return, among others, all the information needed to build a valid signed message. User can call this function only after having opened the session.

Parameters

sess_hdl	handle identifying the active session.
args	pointer to the structure containing the function arguments.

Returns

error code

5.7 SHE commands

Modules

- CMD_ENC_CBC / CMD_DEC_CBC and CMD_ENC_ECB / CMD_DEC_ECB
- CMD_EXPORT_RAM_KEY
- FAST_MAC
- CMD_GENERATE_MAC
- CMD_VERIFY_MAC
- CMD_GET_STATUS
- CMD_LOAD_KEY
- CMD_LOAD_PLAIN_KEY
- CMD_INIT_RNG
- CMD_RND
- CMD_EXTEND_SEED
- last rating code
- CMD_CANCEL

5.7.1 Detailed Description

5.8 CMD_GET_STATUS

Data Structures

- struct op_get_status_args_t

Functions

- she_err_t she_get_status (she_hdl_t utils_handle, op_get_status_args_t ∗args)

5.8.1 Detailed Description

5.8.2 Data Structure Documentation

5.8.2.1 struct op_get_status_args_t

Structure describing the get status operation arguments

Data Fields

uint8_t	sreg	status register bits
uint8_t	pad[3]	padding bytes

5.8.3 Function Documentation

5.8.3.1 she_get_status()

```
she_err_t she_get_status (
        she_hdl_t utils_handle,
        op_get_status_args_t * args )
```

Command to get the content of the status register

Parameters

session_hdl	handle identifying the utils service
args	pointer to the structure containing the function arguments.

Returns

error code.

5.9 Session

Data Structures

- struct she_session_hdl_s
- struct she_service_hdl_s
- struct open_session_args_t

Macros

- #define SHE_HANDLE_NONE (0x0)
- #define SHE_MAX_SESSIONS (8u)

  Maximum sessions supported.
- #define SHE_MAX_SERVICES (32u)

  Maximum services supported.
- #define MAX_KEY_STORE_SESSIONS (5u)

  Maximum Key store sessions supported.
- #define SHE_OPEN_SESSION_PRIORITY_LOW (0x00U)
- #define SHE_OPEN_SESSION_PRIORITY_HIGH (0x01U)
- #define SHE_OPEN_SESSION_FIPS_MODE_MASK BIT(0)
- #define SHE_OPEN_SESSION_EXCLUSIVE_MASK BIT(1)
- #define SHE_OPEN_SESSION_LOW_LATENCY_MASK BIT(3)
- #define SHE_OPEN_SESSION_NO_KEY_STORE_MASK BIT(4)

Typedefs

- typedef uint32_t she_hdl_t

Functions

- struct she_session_hdl_s ∗ she_session_hdl_to_ptr (uint32_t hdl)
- void delete_she_session (struct she_session_hdl_s ∗s_ptr)
- struct she_session_hdl_s ∗ add_she_session (void)
- struct she_service_hdl_s ∗ she_service_hdl_to_ptr (uint32_t hdl)
- void delete_she_service (struct she_service_hdl_s ∗s_ptr)
- struct she_service_hdl_s ∗ add_she_service (struct she_session_hdl_s ∗session)
- she_err_t she_open_session (open_session_args_t ∗args, she_hdl_t ∗session_hdl)
- she_err_t she_close_session (she_hdl_t session_hdl)

5.9.1 Detailed Description

5.9.2 Data Structure Documentation

5.9.2.1 struct she_session_hdl_s Structure describing the session handle members

Data Fields

struct plat_os_abs_hdl ∗	phdl	Pointer to OS device node.
uint32_t	session_hdl	Session handle.
uint32_t	mu_type	Session MU type.
uint32_t	last_rating	last error code returned by command.

5.9.2.2 struct she_service_hdl_s Structure describing the service handle members

Data Fields

struct she_session_hdl_s *	session	Pointer to session handle.
uint32_t	service_hdl	Service handle.

5.9.2.3 struct open_session_args_t Structure detailing the open session operation member arguments

Data Fields

uint32_t	session_hdl	Session handle.
uint8_t	session_priority	Priority of the operations performed in this session.
uint8_t	operating_mode	Options for the session to be opened (bitfield).
uint8_t	interrupt_idx	Interrupt number of the MU used to indicate data availability.
uint8_t	mu_id	index of the MU as per PLAT point of view.
uint8_t	tz	indicate if current partition has TZ enabled.
uint8_t	did	DID of the calling partition.

5.9.3 Macro Definition Documentation

5.9.3.1 SHE_HANDLE_NONE `#define SHE_HANDLE_NONE (0x0)`

Handle not available

5.9.3.2 SHE_OPEN_SESSION_PRIORITY_LOW `#define SHE_OPEN_SESSION_PRIORITY_LOW (0x00U)`

Session opening priority flags Low priority. default setting on platforms that doesn't support sessions priorities.

5.9.3.3 SHE_OPEN_SESSION_PRIORITY_HIGH `#define SHE_OPEN_SESSION_PRIORITY_HIGH (0x01U)`

High Priority session.

5.9.3.4 SHE_OPEN_SESSION_FIPS_MODE_MASK `#define SHE_OPEN_SESSION_FIPS_MODE_MASK BIT(0)`

Operating Mode Only FIPS certified operations authorized in this session.

5.9.3.5 SHE_OPEN_SESSION_EXCLUSIVE_MASK `#define SHE_OPEN_SESSION_EXCLUSIVE_MASK BIT(1)`

No other SHE session will be authorized on the same security enclave.

5.9.3.6 SHE_OPEN_SESSION_LOW_LATENCY_MASK `#define SHE_OPEN_SESSION_LOW_LATENCY_MA←↩ SK BIT(3)`

Use a low latency SHE implementation.

5.9.3.7 SHE_OPEN_SESSION_NO_KEY_STORE_MASK `#define SHE_OPEN_SESSION_NO_KEY_STORE_MA←↩ SK BIT(4)`

No key store will be attached to this session. May provide better performances on some operation depending on the implementation. Usage of the session will be restricted to operations that doesn't involve secret keys (e.g. hash, signature verification, random generation)

5.9.4 Typedef Documentation

5.9.4.1 she_hdl_t `typedef uint32_t she_hdl_t`

Define the SHE handle type

5.9.5 Function Documentation

5.9.5.1 she_session_hdl_to_ptr() `struct she_session_hdl_s* she_session_hdl_to_ptr (`

```
uint32_t hdl )
```

Returns pointer to the session handle

Parameters

hdl	identifying the session handle.

Returns

pointer to the session handle.

5.9.5.2 delete_she_session() `void delete_she_session (`

```
struct she_session_hdl_s * s_ptr )
```

Delete the session

Parameters

s_ptr	pointer identifying the session.

5.9.5.3 add_she_session() `struct she_session_hdl_s* add_she_session ( void )`

Add the session

Returns

pointer to the session.

5.9.5.4 she_service_hdl_to_ptr() `struct she_service_hdl_s* she_service_hdl_to_ptr ( uint32_t hdl )`

Returns pointer to the service handle

Parameters

hdl	identifying the session handle.

Returns

pointer to the service handle.

5.9.5.5 delete_she_service() `void delete_she_service ( struct she_service_hdl_s * s_ptr )`

Delete the service

Parameters

s_ptr	pointer identifying the service.

5.9.5.6 add_she_service() `struct she_service_hdl_s* add_she_service ( struct she_session_hdl_s * session )`

Add the service

Returns

pointer to the service.

5.9.5.7 she_open_session()

```
she_err_t she_open_session (
        open_session_args_t * args,
        she_hdl_t * session_hdl )
```

Parameters

args	pointer to the structure containing the function arguments.
session_hdl	pointer to where the session handle must be written.

Returns

error code.

5.9.5.8 she_close_session()

```
she_err_t she_close_session (
        she_hdl_t session_hdl )
```

Terminate a previously opened session. All the services opened under this session are closed as well

Parameters

session_hdl	pointer to the handle identifying the session to be closed.

Returns

error code.

5.10 Key store

User must open a key store service flow in order to perform the following operations:

Data Structures

- struct open_svc_key_store_args_t
- struct op_key_store_reprov_en_args_t

Macros

- #define KEY_STORE_OPEN_FLAGS_DEFAULT 0x0u
- #define KEY_STORE_OPEN_FLAGS_CREATE 0x1u
- #define KEY_STORE_OPEN_FLAGS_SHE 0x2u
- #define KEY_STORE_OPEN_FLAGS_SET_MAC_LEN 0x8u
- #define KEY_STORE_OPEN_FLAGS_STRICT_OPERATION 0x80u
- #define SHE_STORAGE_CREATE_SUCCESS 0u
- #define SHE_STORAGE_CREATE_WARNING 1u
- #define SHE_STORAGE_CREATE_UNAUTHORIZED 2u
- #define SHE_STORAGE_CREATE_FAIL 3u
- #define SHE_STORAGE_NUMBER_UPDATES_DEFAULT 300u
- #define SHE_STORAGE_MIN_MAC_BIT_LENGTH_DEFAULT 32u

Functions

- she_err_t she_open_key_store_service (she_hdl_t session_hdl, open_svc_key_store_args_t ∗args)
- she_err_t she_close_key_store_service (she_hdl_t key_store_handle)

5.10.1 Detailed Description

User must open a key store service flow in order to perform the following operations:

- create a new key store
- perform operations involving keys stored in the key store (ciphering, signature generation...)
- perform a key store reprovisioning using a signed message. A key store re-provisioning results in erasing all the key stores handled by the SHE.

To grant access to the key store, the caller is authenticated against the domain ID (DID) and Messaging Unit used at the keystore creation, additionally an authentication nonce can be provided.

5.10.2 Data Structure Documentation

5.10.2.1 struct open_svc_key_store_args_t Structure specifying the open key store service member arguments

Data Fields

uint32_t	key_store_hdl	handle identifying the key store service flow
uint32_t	key_store_identifier	user defined id identifying the key store. Only one key store service can be opened on a given key_store_identifier.
uint32_t	authentication_nonce	user defined nonce used as authentication proof for accessing the key store.
uint8_t	flags	bitmap specifying the services properties.
uint16_t	max_updates_number	maximum number of updates authorized for the key store. • Valid only for create operation. • This parameter has the goal to limit the occupation of the monotonic counter used as anti-rollback protection. • If the maximum number of updates is reached, HSM still allows key store updates but without updating the monotonic counter giving the opportunity for rollback attacks.
uint8_t	min_mac_length	it corresponds to the minimum mac length (in bits) accepted to perform MAC verification operations. Only used upon key store creation when KEY_STORE_FLAGS_SET_MAC_LEN bit is set. It is effective only for MAC verification operations with the mac length expressed in bits. It can be used to replace the default value (32 bits). It impacts all MAC algorithms and all key lengths. It must be different from 0. When in FIPS approved mode values $<$ 32 bits are not allowed. Only used on devices implementing SECO FW.
uint8_t ∗	signed_message	pointer to signed_message to be sent only in case of key store re-provisioning.
uint16_t	signed_msg_size	size of the signed_message to be sent only in case of key store re-provisioning.

5.10.2.2 struct op_key_store_reprov_en_args_t Structure describing the key store reprovisioning enable operation arguments

Data Fields

uint8_t ∗	signed_message	signed content payload
uint32_t	signed_msg_size	signed content payload size in bytes

5.10.3 Macro Definition Documentation

5.10.3.1 KEY_STORE_OPEN_FLAGS_DEFAULT `#define KEY_STORE_OPEN_FLAGS_DEFAULT 0x0u`

default flags

5.10.3.2 KEY_STORE_OPEN_FLAGS_CREATE `#define KEY_STORE_OPEN_FLAGS_CREATE 0x1u`

Create a key store

5.10.3.3 KEY_STORE_OPEN_FLAGS_SHE `#define KEY_STORE_OPEN_FLAGS_SHE 0x2u`

Target key store is a SHE key store

5.10.3.4 KEY_STORE_OPEN_FLAGS_SET_MAC_LEN `#define KEY_STORE_OPEN_FLAGS_SET_MAC_L↩ EN 0x8u`

Check min mac length

5.10.3.5 KEY_STORE_OPEN_FLAGS_STRICT_OPERATION `#define KEY_STORE_OPEN_FLAGS_STRICT_O↩ PERATION 0x80u`

The request is completed only when the key store has been written in the NVM and the monotonic counter has been updated. This flag is applicable for CREATE operation only

5.10.3.6 SHE_STORAGE_CREATE_SUCCESS `#define SHE_STORAGE_CREATE_SUCCESS 0u`

New storage created successfully.

5.10.3.7 SHE_STORAGE_CREATE_WARNING `#define SHE_STORAGE_CREATE_WARNING 1u`

New storage created but its usage is restricted to limited security state of chip.

5.10.3.8 SHE_STORAGE_CREATE_UNAUTHORIZED `#define SHE_STORAGE_CREATE_UNAUTHORIZED 2u`

Creation of the storage is not authorized.

5.10.3.9 SHE_STORAGE_CREATE_FAIL `#define SHE_STORAGE_CREATE_FAIL 3u`

Creation of the storage failed for any other reason.

5.10.3.10 SHE_STORAGE_NUMBER_UPDATES_DEFAULT `#define SHE_STORAGE_NUMBER_UPDATES_DEFA↩ ULT 300u`

default number of maximum number of updated for SHE storage.

5.10.3.11 SHE_STORAGE_MIN_MAC_BIT_LENGTH_DEFAULT `#define SHE_STORAGE_MIN_MAC_BIT_LEN↩ GTH_DEFAULT 32u`

default MAC verification length in bits

5.10.4 Function Documentation

5.10.4.1 she_open_key_store_service()

```
she_err_t she_open_key_store_service (
        she_hdl_t session_hdl,
        open_svc_key_store_args_t * args )
```

Open a service flow on the specified key store.

Parameters

session_hdl	SHE handle identifying the current session.
args	pointer to the structure containing the function arguments.

Returns

error code.

5.10.4.2 she_close_key_store_service()

```
she_err_t she_close_key_store_service (
        she_hdl_t key_store_handle )
```

Terminate a previously opened key store service flow

Parameters

key_store_handle	handle identifying the key store service.

Returns

error code.

5.11 CMD_LOAD_KEY

Data Structures

- struct op_key_update_args_t
- struct op_key_update_ext_args_t

Macros

- #define **SHE_LOAD_KEY_EXT_FLAGS_STRICT_OPERATION** BIT(7)

Functions

- she_err_t she_key_update (she_hdl_t utils_handle, op_key_update_args_t ∗args)
- she_err_t she_key_update_ext (she_hdl_t utils_handle, op_key_update_ext_args_t ∗args)

5.11.1 Detailed Description

5.11.2 Data Structure Documentation

5.11.2.1 struct op_key_update_args_t

Structure describing the key update operation arguments

Data Fields

uint32_t	utils_handle	Handle to utils service.
uint32_t	key_ext	identifier of the key extension to be used for the operation
uint32_t	key_id	identifier of the key to be used for the operation
uint8_t ∗	m1	pointer to M1 message
uint8_t	m1_size	size of M1 message - 128 bits
uint8_t ∗	m2	pointer to M2 message
uint8_t	m2_size	size of M2 message - 256 bits
uint8_t ∗	m3	pointer to M3 message
uint8_t	m3_size	size of M3 message - 128 bits
uint8_t ∗	m4	pointer to the output address for M4 message
uint8_t	m4_size	size of M4 message - 256 bits
uint8_t ∗	m5	pointer to the output address for M5 message
uint8_t	m5_size	size of M5 message - 128 bits

5.11.2.2 struct op_key_update_ext_args_t

Structure describing the key update extension operation arguments

Data Fields

uint32_t	utils_handle	Handle to utils service.
uint32_t	key_ext	identifier of the key extension to be used for the operation
uint32_t	key_id	identifier of the key to be used for the operation
uint8_t ∗	m1	pointer to M1 message

Data Fields

uint8_t	m1_size	size of M1 message - 128 bits
uint8_t *	m2	pointer to M2 message
uint8_t	m2_size	size of M2 message - 256 bits
uint8_t *	m3	pointer to M3 message
uint8_t	m3_size	size of M3 message - 128 bits
uint8_t *	m4	pointer to the output address for M4 message
uint8_t	m4_size	size of M4 message - 256 bits
uint8_t *	m5	pointer to the output address for M5 message
uint8_t	m5_size	size of M5 message - 128 bits
uint8_t	flags	bitmap specifying the operations property

5.11.3 Function Documentation

5.11.3.1 she_key_update()

```
she_err_t she_key_update (
            she_hdl_t utils_handle,
            op_key_update_args_t * args )
```

Update an internal key of SHE with the protocol specified by SHE. The request is completed only when the new key has been written in the NVM. The monotonic counter is incremented for each successful update.

Parameters

utils_handle	handle identifying the utils service.
args	pointer to the structure containing the function arguments.

Returns

error code

5.11.3.2 she_key_update_ext()

```
she_err_t she_key_update_ext (
            she_hdl_t utils_handle,
            op_key_update_ext_args_t * args )
```

This is an extension of the CMD_LOAD_KEY The functionality of the CMD_LOAD_KEY is extended by adding a flag argument The updates to the key store must be considered as effective only after an operation specifying the flag "STRICT OPERATION" is aknowledged by SHE

The request is completed only when the key store is written in the NVM and the monotonic counter is incremented

Parameters

utils_handle	handle identifying the utils service
args	pointer to the structure containing the function arguments.

Returns

error code

5.12 CMD_LOAD_PLAIN_KEY

Data Structures

- struct op_load_plain_key_args_t

Functions

- she_err_t she_load_plain_key (she_hdl_t utils_handle, op_load_plain_key_args_t *args)

5.12.1 Detailed Description

5.12.2 Data Structure Documentation

5.12.2.1 struct op_load_plain_key_args_t Structure describing the plain key load operation arguments

Data Fields

uint8_t	key[SHE_KEY_SIZE_IN_BYTES]	pointer to plain key

5.12.3 Function Documentation

5.12.3.1 she_load_plain_key()

```
she_err_t she_load_plain_key (
            she_hdl_t utils_handle,
            op_load_plain_key_args_t * args )
```

Load a key as plaintext to the RAM_KEY slot without encryption and verification.

Parameters

hdl	pointer to the SHE utils handle
key	pointer to the plaintext key to be loaded - 128bits

Returns

error code

5.13 CMD_INIT_RNG

Functions

- she_err_t she_open_rng_service (she_hdl_t session_hdl, open_svc_rng_args_t ∗args)
- she_err_t she_close_rng_service (she_hdl_t rng_handle)

5.13.1 Detailed Description

5.13.2 Function Documentation

5.13.2.1 she_open_rng_service()

```
she_err_t she_open_rng_service (
            she_hdl_t session_hdl,
            open_svc_rng_args_t ∗ args )
```

initializes the seed and derives a key for the PRNG. The function must be called before CMD_RND after every power cycle/reset.

User can call this function only after having opened a session.

Parameters

session_hdl	handle identifying the current session.
args	pointer to the structure containing the function arguments.

Returns

error code

5.13.2.2 she_close_rng_service()

```
she_err_t she_close_rng_service (
            she_hdl_t rng_handle )
```

Terminate a previously opened rng service flow

Parameters

rng_handle	handle identifying the RNG session.

Returns

error code

5.14 CMD_RND

Data Structures

- struct open_svc_rng_args_t
- struct op_get_random_args_t

Macros

- #define SHE_RND_SIZE 16u

Typedefs

- typedef uint8_t **svc_rng_flags_t**

Functions

- she_err_t she_get_random (she_hdl_t rng_handle, op_get_random_args_t ∗args)

5.14.1 Detailed Description

5.14.2 Data Structure Documentation

Data Fields

svc_rng_flags_t	flags	bitmap indicating the service flow properties
uint8_t	reserved[3]	
uint32_t	rng_hdl	rng handle

5.14.2.1 struct open_svc_rng_args_t

5.14.2.2 struct op_get_random_args_t

Structure detailing the get random number operation member arguments

Data Fields

uint8_t ∗	output	pointer to the output area where the random number must be written
uint32_t	random_size	length in bytes of the random number to be provided.
svc_rng_flags_t	svc_flags	bitmap indicating the service flow properties
uint8_t	reserved[3]	

5.14.3 Macro Definition Documentation

5.14.3.1 SHE_RND_SIZE `#define SHE_RND_SIZE 16u`

size of random data for SHE

5.14.4 Function Documentation

5.14.4.1 she_get_random()

```
she_err_t she_get_random (
            she_hdl_t rng_handle,
            op_get_random_args_t * args )
```

returns a vector of 128 random bits. The random number generator has to be initialized by CMD_INIT_RNG before random numbers can be supplied.

Parameters

rng_handle	handle identifying the RNG service
args	pointer to the structure containing the function arguments.

Returns

error code

5.15 CMD_EXTEND_SEED

Data Structures

- struct op_rng_extend_seed_t

Macros

- #define SHE_ENTROPY_SIZE 16u

Functions

- she_err_t she_extend_seed (she_hdl_t rng_handle, op_rng_extend_seed_t ∗args)

5.15.1 Detailed Description

5.15.2 Data Structure Documentation

5.15.2.1 struct op_rng_extend_seed_t Structure describing the RNG extend seed operation arguments

Data Fields

uint32_t	entropy[4]	< entropy to extend seed entropy size
uint32_t	entropy_size	

5.15.3 Macro Definition Documentation

5.15.3.1 SHE_ENTROPY_SIZE `#define SHE_ENTROPY_SIZE 16u`

size of entropy for SHE

5.15.4 Function Documentation

5.15.4.1 she_extend_seed()

```
she_err_t she_extend_seed (
        she_hdl_t rng_handle,
        op_rng_extend_seed_t * args )
```

extends the seed of the PRNG by compressing the former seed value and the supplied entropy into a new seed which will be used to generate the following random numbers. The random number generator has to be initialized by CMD_INIT_RNG before the seed can be extended.

Parameters

rng_handle	handle identifying the RNG service
args	pointer to the structure containing entropy vector (128bits)

Returns

error code

5.16 Shared Buffer

Data Structures

- struct op_shared_buf_args_t
- struct open_svc_cipher_args_t
- struct op_cipher_one_go_args_t

5.16.1 Detailed Description

5.16.2 Data Structure Documentation

5.16.2.1 struct op_shared_buf_args_t Structure describing the get shared buffer operation arguments

Data Fields

uint16_t	shared_buf_offset	offset of the shared buffer in secure memory
uint16_t	shared_buf_size	size in bytes of the allocated shared buffer

5.16.2.2 struct open_svc_cipher_args_t Structure describing the open cipher service members

Data Fields

uint32_t	cipher_hdl	handle identifying the cipher service flow
uint8_t	flags	bitmap specifying the services properties
uint8_t	reserved[3]	

5.16.2.3 struct op_cipher_one_go_args_t Structure describing the cipher one go operation arguments

Data Fields

uint32_t	key_identifier	identifier of the key to be used for the operation
uint8_t ∗	iv	pointer to the initialization vector (nonce in case of AES CCM)
uint16_t	iv_size	length in bytes of the initialization vector. it must be 0 for algorithms not using the initialization vector. It must be 12 for AES in CCM mode
uint8_t	svc_flags	bitmap specifying the services properties.
uint8_t	flags	bitmap specifying the operation attributes
uint8_t	cipher_algo	algorithm to be used for the operation
uint8_t ∗	input	pointer to the input area: • plaintext for encryption • ciphertext for decryption Note: In case of CCM it is the purported ciphertext.
uint8_t ∗	output	pointer to the output area: • ciphertext for encryption Note: In case of CCM it is the output of the generation-encryption process. • plaintext for decryption

Data Fields

uint32_t	input_size	length in bytes of the input. • In case of CBC and ECB, the input size should be multiple of a block cipher size (16 bytes).
uint32_t	output_size	length in bytes of the output

5.17 Error codes

Error codes returned by SHE functions.

Enumerations

- enum she_err_t {
  SHE_NO_ERROR = 0x0,
  SHE_SEQUENCE_ERROR = 0x1,
  SHE_KEY_NOT_AVAILABLE = 0x2,
  SHE_KEY_INVALID = 0x3,
  SHE_KEY_EMPTY = 0x4,
  SHE_NO_SECURE_BOOT = 0x5,
  SHE_KEY_WRITE_PROTECTED = 0x6,
  SHE_KEY_UPDATE_ERROR = 0x7,
  SHE_RNG_SEED = 0x8,
  SHE_NO_DEBUGGING = 0x9,
  SHE_BUSY = 0xA,
  SHE_MEMORY_FAILURE = 0xB,
  SHE_GENERAL_ERROR = 0xC,
  SHE_UNKNOWN_WARNING = 0x27,
  SHE_FATAL_FAILURE = 0x29,
  SHE_LIB_ERROR = 0xEF }

5.17.1 Detailed Description

Error codes returned by SHE functions.

5.17.2 Enumeration Type Documentation

5.17.2.1 she_err_t `enum she_err_t`

Error codes returned by SHE functions.

Enumerator

SHE_NO_ERROR	Success.
SHE_SEQUENCE_ERROR	Invalid sequence of commands.
SHE_KEY_NOT_AVAILABLE	Key is locked.
SHE_KEY_INVALID	Key not allowed for the given operation.
SHE_KEY_EMPTY	Key has not beed initialized yet.
SHE_NO_SECURE_BOOT	Conditions for secure boot process are not met.
SHE_KEY_WRITE_PROTECTED	Memory slot for key has been write-protected.
SHE_KEY_UPDATE_ERROR	Key update failed due to errors in verification of the messages.
SHE_RNG_SEED	The seed has not been initialized.
SHE_NO_DEBUGGING	Internal debugging is not possible.
SHE_BUSY	A function of SHE is called while another function is still processing.
SHE_MEMORY_FAILURE	Memory error (e.g. flipped bits).
SHE_GENERAL_ERROR	Error not covered by other codes occurred.
SHE_UNKNOWN_WARNING	SHE Unknown Warning.
SHE_FATAL_FAILURE	A fatal failure occurred, SHE goes in unrecoverable error state not replying to further requests
SHE_LIB_ERROR	SHE lib

5.18 Utils

User must open a SHE utils service flow in order to perform the following operations:

Data Structures

- struct op_open_utils_args_t

Functions

- she_err_t she_open_utils (she_hdl_t key_store_handle, op_open_utils_args_t ∗args)
- she_err_t she_close_utils (she_hdl_t utils_handle)

5.18.1 Detailed Description

User must open a SHE utils service flow in order to perform the following operations:

- Create a utils handle
- perform SHE key update extension
- update SHE plain key
- export SHE plain key
- get SHE identity (UID)
- get SHE status register
- perform MAC generation and verification in fast mode for a SHE session on V2X
- perform MAC generation and verification in fast mode for a SHE session

5.18.2 Data Structure Documentation

5.18.2.1 struct op_open_utils_args_t Structure describing the open utils service operation arguments

Data Fields

uint32_t	utils_handle	

5.18.3 Function Documentation

5.18.3.1 she_open_utils()

```
she_err_t she_open_utils (
        she_hdl_t key_store_handle,
        op_open_utils_args_t * args )
```

Open SHE utils service flow on the specified key store. The SHE utils service flow can be opened only after opening SHE key storage handle.

Parameters

key_store_handle	handle identifying the key store service.
args	pointer to the structure containing the function arguments.

Returns

error code.

5.18.3.2 she_close_utils()

```
she_err_t she_close_utils (
        she_hdl_t utils_handle )
```

Terminate a previously opened utils service flow

Parameters

utils_handle	handle identifying the utils service.

Returns

error code.

5.19 last rating code

Functions

- uint32_t she_get_last_rating_code (she_hdl_t session_hdl)

5.19.1 Detailed Description

5.19.2 Function Documentation

5.19.2.1 she_get_last_rating_code() `uint32_t she_get_last_rating_code ( she_hdl_t session_hdl )`

Report rating code from last command

SHE API defines standard errors that should be returned by API calls. Error code reported by SECO are "translated" to these SHE error codes. This API allow user to get the error code reported by SECO for the last command before its translation to SHE error codes. This shoudl be used for debug purpose only.

Parameters

session_hdl	SHE session handler

Returns

rating code reported by last command

5.20 CMD_CANCEL

Functions

- void she_cmd_cancel (void)

5.20.1 Detailed Description

5.20.2 Function Documentation

5.20.2.1 she_cmd_cancel()

```
void she_cmd_cancel (
    void  )
```

interrupt any given function and discard all calculations and results.

5.21 Get Info

Data Structures

- struct op_get_info_args_t

5.21.1 Detailed Description

5.21.2 Data Structure Documentation

5.21.2.1 struct op_get_info_args_t

Structure describing the get info operation member arguments

Data Fields

uint32_t	user_sab_id	Stores User identifier (32bits)
uint8_t *	chip_unique_id	Stores the chip unique identifier.
uint16_t	chip_unq_id_sz	Size of the chip unique identifier in bytes.
uint16_t	chip_monotonic_counter	Stores the chip monotonic counter value (16bits)
uint16_t	chip_life_cycle	Stores the chip current life cycle bitfield (16bits)
uint32_t	version	Stores the module version (32bits)
uint32_t	version_ext	Stores the module extended version (32bits)
uint8_t	fips_mode	Stores the FIPS mode bitfield (8bits). Bitmask definition: bit0 - FIPS mode of operation: • value 0 - part is running in FIPS non-approved mode. • value 1 - part is running in FIPS approved mode. bit1 - FIPS certified part: • value 0 - part is not FIPS certified. • value 1 - part is FIPS certified. bit2-7: reserved • value 0.

5.22 Global Info

Macros

- #define **SOC_IMX8DXL** 0xe
- #define **SOC_IMX8ULP** 0x84d
- #define **SOC_IMX93** 0x9300
- #define **SOC_IMX95** 0x9500
- #define **SOC_REV_A0** 0xa000
- #define **SOC_REV_A1** 0xa100
- #define **SOC_REV_A2** 0xa200
- #define **SOC_REV_B0** 0xb000
- #define **SOC_LF_FAB_DEFAULT** 0x1
- #define **SOC_LF_FAB_MODE** 0x2
- #define **SOC_LF_NO_NXP_SECRETS** 0x4
- #define **SOC_LF_WITH_NXP_SECRETS** 0x8
- #define **SOC_LF_SCU_FW_CLOSED** 0x10
- #define **SOC_LF_SECO_FW_CLOSED** 0x20
- #define **SOC_LF_CLOSED** 0x40
- #define **SOC_LF_CLOSED_WITH_NXP_FW** 0x80
- #define **SOC_LF_PARTIAL_FIELD_RET** 0x100
- #define **SOC_LF_FIELD_RET** 0x200
- #define **SOC_LF_NO_RET** 0x400
- #define **GINFO_LIB_VERSION_LEN** 16
- #define **GINFO_NVM_VERSION_LEN** 16
- #define **GINFO_COMMIT_ID_SZ** 40
- #define **HSM_API_VERSION_1** 0x1
- #define **HSM_API_VERSION_2** 0x2

Functions

- void populate_global_info (hsm_hdl_t hsm_session_hdl)
- void show_global_info (void)
- bool is_global_info_populated (void)
- uint16_t se_get_soc_id (void)
- uint16_t se_get_soc_rev (void)
- uint16_t se_get_chip_lifecycle (void)
- uint8_t se_get_fips_mode (void)
- uint8_t se_get_lib_newness_ver (void)
- uint8_t se_get_lib_major_ver (void)
- uint8_t se_get_lib_minor_ver (void)
- uint8_t se_get_nvm_newness_ver (void)
- uint8_t se_get_nvm_major_ver (void)
- uint8_t se_get_nvm_minor_ver (void)
- const char ∗ se_get_commit_id (void)
- const char ∗ se_get_lib_version (void)
- const char ∗ se_get_nvm_version (void)
- const char ∗ get_soc_id_str (uint16_t soc_id)
- const char ∗ get_soc_rev_str (uint16_t soc_rev)
- const char ∗ get_soc_lf_str (uint16_t lifecycle)
- void se_get_info (uint32_t session_hdl, op_get_info_args_t ∗args)
- void se_get_soc_info (uint32_t session_hdl, uint16_t ∗soc_id, uint16_t ∗soc_rev)

5.22.1 Detailed Description

5.22.2 Function Documentation

5.22.2.1 populate_global_info()

```
void populate_global_info (
        hsm_hdl_t hsm_session_hdl )
```

Populate the Global Info structure

Parameters

hsm_session_hdl	identifying the session.

5.22.2.2 show_global_info()

```
void show_global_info (
        void )
```

Print the Global Info of library

5.22.2.3 is_global_info_populated()

```
bool is_global_info_populated (
        void )
```

Get the status of Global Info, if populated or not.

5.22.2.4 se_get_soc_id()

```
uint16_t se_get_soc_id (
        void )
```

Get SoC ID.

5.22.2.5 se_get_soc_rev()

```
uint16_t se_get_soc_rev (
        void )
```

Get SoC Revision.

5.22.2.6 se_get_chip_lifecycle()

```
uint16_t se_get_chip_lifecycle (
        void )
```

Get Chip-lifecycle.

5.22.2.7 se_get_fips_mode()

```
uint8_t se_get_fips_mode (
        void )
```

Get Fips mode.

5.22.2.8 se_get_lib_newness_ver() `uint8_t se_get_lib_newness_ver ( void )`

Get library newness version.

5.22.2.9 se_get_lib_major_ver() `uint8_t se_get_lib_major_ver ( void )`

Get library major version.

5.22.2.10 se_get_lib_minor_ver() `uint8_t se_get_lib_minor_ver ( void )`

Get library minor version.

5.22.2.11 se_get_nvm_newness_ver() `uint8_t se_get_nvm_newness_ver ( void )`

Get NVM newness version.

5.22.2.12 se_get_nvm_major_ver() `uint8_t se_get_nvm_major_ver ( void )`

Get NVM major version.

5.22.2.13 se_get_nvm_minor_ver() `uint8_t se_get_nvm_minor_ver ( void )`

Get NVM minor version.

5.22.2.14 se_get_commit_id() `const char* se_get_commit_id ( void )`

Get Build commit id.

5.22.2.15 se_get_lib_version() `const char* se_get_lib_version ( void )`

Get library version string.

5.22.2.16 se_get_nvm_version() `const char* se_get_nvm_version ( void )`

Get NVM version string.

5.22.2.17 get_soc_id_str() `const char* get_soc_id_str ( uint16_t soc_id )`

Get the string representating SoC ID

Parameters

soc↩ _id	SoC ID fetched from Global Info

Returns

String represention of the SoC ID

5.22.2.18 get_soc_rev_str()

```
const char* get_soc_rev_str (
            uint16_t soc_rev )
```

Get the string representating SoC Revision

Parameters

soc_rev	SoC Revision fetched from Global Info

Returns

String represention of the SoC Revision

5.22.2.19 get_soc_lf_str()

```
const char* get_soc_lf_str (
            uint16_t lifecycle )
```

Get the string representation of the Chip Lifecycle

Parameters

lifecycle	value fetched from Global Info

Returns

a string represention of Lifecycle

5.22.2.20 se_get_info()

```
void se_get_info (
            uint32_t session_hdl,
            op_get_info_args_t * args )
```

Get Info for Global Info setup

5.22.2.21 se_get_soc_info()

```
void se_get_soc_info (
            uint32_t session_hdl,
            uint16_t * soc_id,
            uint16_t * soc_rev )
```

Get SoC Info for Global Info setup

Index